# Implementation of Robust Predictive Controller on FPGA device

TELMOUDI BRINI Sirine[1], BOUZOUITA Badreddine[2], BOUANI Faouzi[1]
[1] Tunis El Manar University, National Engineering School of Tunis, Tunisia,
LR11ES20 Laboratory of Analysis Conception and Control of Systems, Tunis, Tunisia.
[2] University of Sousse, National Engineering School of Sousse, Tunisia
Sirinebrini@gmail.com, badreddine.bouzouita@enit.rnu.tn, faouzi.bouani@enit.rnu.tn

*Abstract*— **Implementation of a linear Robust Predictive Controller (RPC) into a FPGA with 20 MHz fixed clock is presented in this work. The design of the controller is based on a single input-single output Auto Regressive Integrated Moving Average (ARIMA) Model. To take into account the uncertain systems behaviour, the parametric polytopic uncertainties are adopted. Supported on worst case strategy, by the resolution of a min-max optimization problem the control law is obtained. Since the performance criterion to be optimized is non-convex, a non-determinist global optimization method based on Genetic Algorithms (GA) is proposed to solve this problem. The efficiency of the proposed approach is demonstrated with the RPC implementation on a Nanoboard 3000XN FPGA platform chip using the Altium Designer such a conception environment.**

***Keywords***— FPGA, Robust Predictive Control, global optimization, Genetic Algorithm, Altium Designer.

## I. Introduction

Some control technique requires a numerical model intending to predict the future behavior of the system [1]. Thereby, the precision degrees of the considered model cheek a significant role on the controller effectiveness. Therefore, it is necessary to use a controller which ensures the desired performances in the presence of uncertainties [9]. In this case, we talk about robust control. Certainly, predictive control took advantage to exploit systems power of embarked systems on the control processes and also the evolution of the optimization algorithms which became faster [2].

In real-time applications managed using intelligent technology, with the technological advancement in the field of microelectronics, new hardware design solutions such as Field Programmable Gate Array (FPGA), Application Specific Integrated Circuit (ASICs) are available and can be used as digital targets for implementation of the control algorithms in a single component [7], [8]. The advantages of such an implementation are multiple: the reduction of the execution time by adopting parallel processing, the rapid prototyping of the numerical control on FPGA, the confidentiality of the architecture [13], the possibility of applying intelligent commands that have recourse to heavier techniques in terms of computing time and improving the quality of control of industrial process by exploiting the new digital systems technologies [17]. Nowadays, all of these advantages form a need and a necessity for the control of industrial systems characterized by high performance [15]. The concept of robustness has emerged to handle a set of analysis and synthesis problems especially for systems, whose models aren't precisely known [12]. When the structure of the model is uncertain, by adopting the worst case strategy, the calculation of the robust predictive control law amounts to minimizing the maximum of a criterion with respect to the control signal, taking into account all the possible models described by the set of uncertainties such as parametric uncertainties and polytopic uncertainties [5]. Therefore, the control law is obtained by solving a min-max optimization problem [6]. The cost function to be optimized for the robust predictive control is non-convex opposite the uncertainties. Also by using a local optimization technique non optimal control law is obtained. So we proposed global optimization algorithms as genetic algorithms (GA) method [3], [14] which uses stochastic rules and decisions in order to provide the global solution and has known a considerable interest in predictive control.

It is in this spirit that we develop this work which is interested in the synthesis of a robust control law implemented into Nanoboard 3000XN FPGA platform, based on a linear and uncertain ARIMA model which can manipulate unstable open loop systems and decrease the number of model parameters.
The remainder of the paper is organized as follows: section II presents theoretical formulation of RPC. Section III gives the details of RPC algorithm implementation on FPGA. Section IV shows simulation results. The conclusion is provided in section V.

## II. Problem Formulation of RPC

### A. Linear model

In this section, the output predictions are presented by ARIMA model:

$$\Delta y(k) = \frac{B(q^{-1})}{A(q^{-1})}\Delta u(k) \tag{1}$$

where $y(k)$ is the system output and $u(k)$ is the system input. The term $\Delta = 1 - q^{-1}$ is the integral action that allows the cancellation of the static error. $A(q^{-1})$ and $B(q^{-1})$ are polynomials of degrees respectively $na$ and $nb$ in backward shift operator $q^{-1}$ which are bounded and uncertain to take into account the uncertain behavior of the system. These polynomials are given as follows:

$$A\left(q^{-1}\right) = 1 + \sum_{i=1}^{na} a_i q^{-i} \ , \quad a_i \in \left[\underline{a_i}, \overline{a_i}\right], i = 1,\dots,n_a \tag{2}$$

$$B(q^{-1}) = \sum_{j=1}^{nb} b_j q^{-i}, b_j \in \left[\underline{b_j}, \overline{b_j}\right], j = 1,\dots,n_b \tag{3}$$

with $\underline{a}_i$ and $\overline{a}_i$ are respectively the lower and the upper bounds of $a_i$, $\underline{b}_j$ and $\overline{b}_j$ are respectively the lower and the upper bounds of $b_j$.

Therefore, the output prediction $\hat{y}(k+j)$ can be obtained by a multiple recursion using relation (1):

$$\hat{y}(k+j)=\sum_{i=1}^{j} g_{j-i+1}\Delta u(k+i-1)+y_l(k+j) \text{ for } j \geq 1 \tag{4}$$

with:

$$\begin{cases} g_1 = b_1 \\ g_j = b_j + \sum_{i=1}^{j-1}(a_{i-1}-a_i)g_{j-i} \end{cases} \tag{5}$$

and:

$$\begin{cases} y_l(k+1)=y(k)-\sum_{i=1}^{na}a_i\Delta y(k+1-i)+\sum_{i=2}^{nb}b_i\Delta u(k+1-i) \\ y_l(k+j)=a_{j-1}y(k)+\sum_{i=j+1}^{nb}b_i\Delta u(k+j-i)+\sum_{i=1}^{min(j-1,na)}(a_{i-1}-a_i)y_l(k+j-i) \end{cases} \tag{6}$$

*B. Control law*

Robust predictive control requires online optimisation of the following min-max optimisation problem:

$$\min_{\Delta U}\text{-}\max_{\psi} J \tag{7}$$

where $\Delta U$ is the vector of future optimized control increments, $\psi$ represents the set of uncertainties and $J$ denotes the desired performances and presented by the following Quadratic Problem (QP) [23]:

$$J=\sum_{j=N_i}^{N_y}(y_c(k+j)-\hat{y}(k+j))^2+\lambda\sum_{j=0}^{N_u-1}\Delta u(k+j)^2 \tag{8}$$

with $\hat{y}(k+j)$: predicted output. $y_c(k+j)$: set point. $\lambda$: weighting on the future increment control. $\Delta$: incremental operator. $u(k)$: the input control. $N_y, N_u$ and $N_i$ denote the output prediction horizon, control horizon and the initial prediction horizon respectively.

We can solve the optimisation problem (7) in two steps; starting by the calculation of the parameters that maximize the cost function $J$, and next, we minimise it facing the input control:

$$\min_{\Delta U}\text{-}\max_{\psi} J = \min_{\Delta U} J^*(\Delta U) \tag{9}$$

with
$$J^*(\Delta U)=\max_{\psi} J \tag{10}$$

We can show that the cost function $J$ to be optimized is non-convex opposite the adopted uncertainties. Therefore, the optimization problem (10) is non-convex. The use of a local optimization method such as the gradient algorithm leads to non-optimal control law. In the next section, we propose global optimization algorithm to solve this problem.

*C. Genetic Algorithms*

Genetic algorithms (AG) are stochastic optimization algorithms based on the mechanisms of natural selection and genetics [14]. However, the natural processes to which they refer are much more complex. The individual in a population is represented by a chromosome consisting of genes that contain the hereditary characteristics of the individual. The principles of selection, crossing and mutation are inspired by natural processes of the same name.

To use a genetic algorithm on a particular problem, the following elements must be available:

➢ *Codification*:

This step consists to describe each individual by a string of alphabet which can be binary, real, symbol or any other [3]. In this work we have chosen the real codification because our parameters are real. This description is called *chromosome* and a set of chromosome presents a *population.*

➢ *Fitness function:*

The variable to be optimized can be, for example, consumption, yield, transmission factor, profit, cost, development time etc. Thus, an optimization algorithm requires the definition of a function called *fitness function* which allows evaluating of the potential solutions from the variable to be optimized. As a result, the algorithm converges towards an optimum of this function [3].

➢ *Generation of a new population:*

The initial population $pop_0$ is generated randomly, then for each individual the evaluation function $f_i$ is calculated; from it we obtain the new population using the three genetic steps illustrated as follow:

- *Selection*: the aim of selection is to retain the best individuals to ensure the convergence of the algorithm. Tournament selection is an alternative to proportional selection. This step is described in algorithm 1.

**Algorithm 1: selection step of GA**

```
1: Evaluation of population individuals: $f_j$
2: Calculate the total sum of population: $F=\sum_{j=1}^{n} f_j$
   $n$: number of population chromosomes.
3: Calculate the selection probability: $P_j=\frac{f_j}{F}$
4: Calculate the cumulative probability: $q_j = P_1+\cdots+P_j$
5: Beginning selection:
      • Generate randomly $0 \leq r \leq 1$
      • If $r < q_1$ then select the chromosome $P_0$
        • Else
          o for j from 1 to n
            o If $q_{j-1} \leq r \leq q_j$
              o then select the chromosome $P_j$
            o End if
          o End for
      • End if
6: End selection
```

- *Crossover*: the crossing operator is active, with a probability $P_c$, a pair of parent chromosomes $P_1$ and $P_2$ returns a chromosome child $C$ using the arithmetic crossing. The recommended values for the crossover probability parameter are $P_c = 0.95$ from [9] and $P_c \in [0.75, 0.95]$ from [10]. In this step, the population will be divided on two parts, the chromosomes of the first part will be cross two by two, and the chromosome children obtained from each cross will be injected into the second half of the population as it is shown in the algorithm 2. the arithmetic crossing is defined by:

$$C = P_c P_1 + (1 - P_c) P_2 \qquad (11)$$

**Algorithm2: Crossover step of GA**

```
1: Define the crossover probability parameter $P_c$
    • For i from n/2 to n do
          $C(i) = (1 - P_c) P_{n-i+1} + P_c P_{n-i+2}$
    • End for
```

*Mutation:* in order to furnish genetic diversity, with a probability $P_m$, this step enables the spontaneous transform of an individual, which randomly generates directions that are adaptive with respect to the last successful or unsuccessful generation, is considered. This step is described in algorithm 3.

**Algorithm3: Mutation step of GA**

```
1: Define the mutation probability parameter $P_m$
2: Generate randomly $0 < r < 1$ :
    • For i from 0 to n/2 do
          If $r < P_m$ then
        $POP_{next}(i) = POP_0(0)$
        with $POP_{next}$ is the next population.
       End if
    • End for
```

➢ *Stopping conditions:*

Stopping the algorithm is conditioned by the validation of certain criteria which may be the maximum number of generations or when the individuals of a population don't evolve more rapidly or the maximum time of execution or even a combination of these criteria.

## III. IMPLEMENTATION OF LINEAR RPC ON FPGA

This section presents the method to implement the RPC algorithm on the Map Nanoboard 3000XN supporting Xilinx FPGA.

### A. *Nanoboard 3000XN*

The Nanoboard 3000XN map with a fixed Xilinx Spartan-3AN device (XC3S1400AN-4FGG676C) is crucial for the speedy progression of the systems embedded with the Altium Designer. It is a platform of a reprogrammable design material that exploits the power dedicated to the electronic conception of high capacity, allowing the fast implementation and the debugging of our interactive conceptions [16]. This map is constituted by several material modules which realize miscellaneous tasks according to the designer's needs, such an integrated color TFT LCD panel (240x320), an SVGA interface (24-bits, 80 MHz), a programmable clock (6 to 200 MHz) and a fixed one (20 MHz) both available to FPGA users, a 4-channels 8-bits ADC, and 4-channels 8-bits DAC.

### B. *RPC conception*

Fig. 1 presents the solution for the custom hardware. The predictive algorithm is implemented in C++ and compiled for a Nanoboard 3000XN XC3S1400AN-4FGG676C FPGA. In addition, a TFT screen shows the evolution of the output and control action.

- The first step of the prediction process is for a sample cycle $k$ such that the FPGA calculates the output data $y(k)$ of the system.
- The second step consists to maximize the cost function $J$ to estimate the optimal model parameters $a_{i\,op}$ and $b_{j\,op}$ , using the GA such a non determinist optimized method.
- In the next step, the state observer prepares the data calculating the free responses $y_l$ to feed into the genetic algorithm Min-Max-Problem(Min-Max-P) solver.
- Then the Min-Max-P is solved to give a new sequence of the control action. Only one control signal, $u(k)$ is implemented on the plant while the others are rejected, due to the next sampling instant, $y(k+1)$ is known.
- The LCD screen is used to display the evolutions of output and control action signals.
- The first step is repeated with the updated value and all sequences are brought up to date.

***RPC Design***

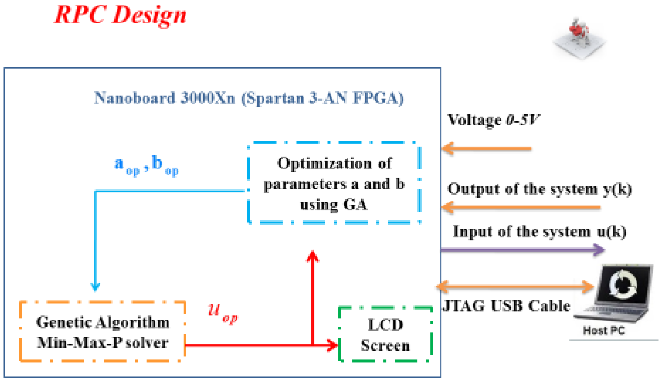
Nanoboard 3000Xn (Spartan 3-AN FPGA)
Voltage 0-5V
$a_{op}, b_{op}$
Optimization of parameters a and b using GA
Output of the system y(k)
Input of the system u(k)
$u_{op}$
Genetic Algorithm Min-Max-P solver
LCD Screen
JTAG USB Cable
Host PC

Fig. 1: Structure of the implemented RPC.

### C. *Hardware architecture*

The model given in Fig. 2 approves the designer to construct a processor-based system with a more rational and abstract approach. In this design, a processor TSK3000 is founded to ensure the execution of a software application, which is a 32-bit RISC processor. Most of its instructions are 32 bits wide and run in a single clock cycle. Moreover, there is a memory SRAM to store the data, a controller TFT_VGA to pilot the display of signals on the LCD screen, and two wishbones (one for interconnection and one for arbitration).

### D. *Software architecture*

The important step in the software process is the choice of libraries, as the absence of one library can generate some errors at compilation, so the user must comprehend the utility of each unit before using it [16]. The LCD module of the card Nanoboard 3000XN is utilized where the function name of each library provided by the map shown in Table I is needed.

In order to use the RPC controller algorithm, the steps illustrated in the flowchart of Fig. 3 must be followed.

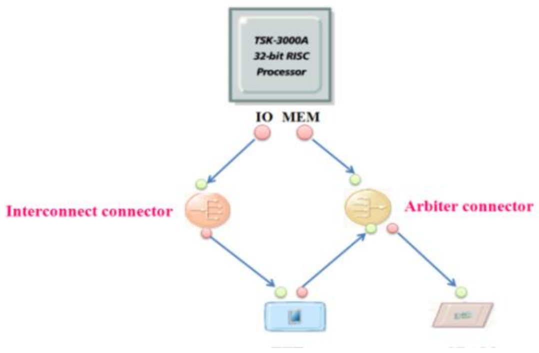

Fig. 2: Hardware architecture for RPC

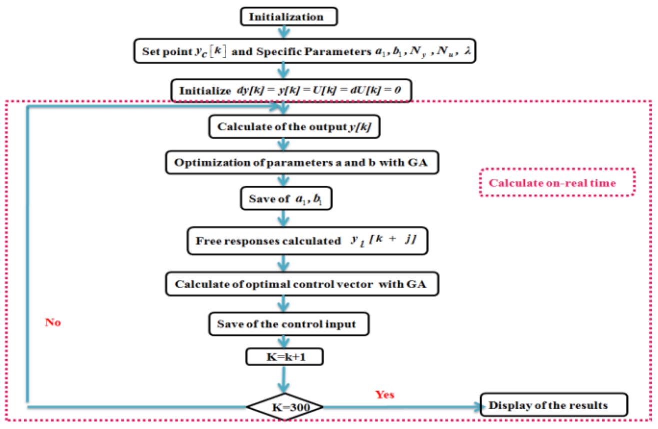
Initialization
Set point $y_c[k]$ and Specific Parameters $a_1, b_1, N_y, N_u, \lambda$
Initialize $dy[k] = y[k] = U[k] = dU[k] = 0$
Calculate of the output $y[k]$
Optimization of parameters a and b with GA
Save of $a_1, b_1$
Free responses calculated $y_l[k+j]$
Calculate of optimal control vector with GA
Save of the control input
K=k+1
K=300
No
Yes
Display of the results
Calculate on-real time

Fig. 3: Flowchart of the robust predictive control algorithm

## IV. Simulation Results

In this section, linear robust predictive controller based on ARIMA model proposed in this work is implemented into the map Nanoboard 3000XN which support an FPGA from the Xilinx family. The RPC scheme is designed with the following parameters:

- ARIMA model: $\Delta y(k) = \dfrac{B(q^{-1})}{A(q^{-1})}\Delta u(k)$

  where : $A(q^{-1}) = 1 + a_1 q^{-1}$, $B(q^{-1}) = b_1 q^{-1}$
- Constraints on input signal: $0v \le u(k) \le 5v$
- Constraints onto parameters $a_1$ and $b_1$:

  $-0.99 \le a_1 \le -0.4$ and $0.4 \le b_1 \le 0.8$
- Fitness function: is defined by the cost function to be optimized.
- Population size: is defined by the parameter *n*. A several test are used to determine the optimal size.
- Stopping criteria: is defined by the parameter *num* and it's specified by the maximum number of generations.
- Crossover and mutation probabilities: $P_m = 0.3$ and $P_C = 0.75$

Two cases are envisaged to test the efficient of the GA.

**Case 1:** $a_1$ and $b_1$ are fixed

We suppose that the system is characterized by the following fixed parameters: $a_1 = -0.97, b_1 = 0.6$ for i=1, .., 100

**Case 2:** $a_1$ and $b_1$ are variables

Assume that the system used is characterized by the following parameters:

$a_1 = -0.97, b_1 = 0.6$ for i=1, .., 30

$a_1 = -0.5, b_1 = 0.4$ for i=31, ..., 70

$a_1 = -0.97, b_1 = 0.6$ for i=71, ..., 100

It should be noted that for the two cases even if the parameters $a_1$, $b_1$ and $u$ are multiplied by the crossover probability parameter in the genetic algorithm, their values always remain in the constraint interval and by the consequence children are always feasible with respect to linear constraints and bounds.

### A. *Model parameters estimation*

To solve the optimisation problem (7), calculation of the parameters that maximize the cost function $J$ is required. In this step the genetic algorithm is used as a global optimization method to find, at each iteration, the model parameters. The size of the initial population is $\dim(P_0) = (num, 3)$ and it's defined as follows:

$$P_0 = \begin{pmatrix} a_{11} & b_{11} & u_1 \\ & \cdots & \\ a_{1num} & b_{1num} & u_{num} \end{pmatrix}$$

The first and the second columns are reserved to the model parameters and the third one is dedicated to the control input. For each population individual $ind_i = (a_{1i}, b_{1i}, u_i), i = 1, \ldots, num$, we compute the prediction sequence $\hat{y}(k+j)$, $j = 1 \ldots N_y$ and we evaluate the performance criterion *J* given by (8). Based on the criterion values (fitness), the GA operators (selection, crossover and mutation) are applied in order to find the next population individuals. The procedure will be repeated until a stopping criterion is reached. Then, the parameters of the worst case model are obtained.

For the second optimization problem, we use the following initial population:

$$P_{02} = \begin{pmatrix} u_1 \\ \cdots \\ u_{num} \end{pmatrix}$$

In this case, the size of the initial population is $(num, 1)$ and the model parameters are those obtained in the first optimisation problem. The evolution of the estimated model parameters is illustrated in Fig. 4 (case 1) and Fig.5 (case 2), it can be seen that for the two cases the values of $a_1$ and $b_1$ are always remain in the constraint interval.

TABLE I
FUNCTION NAME OF EACH LIBRARY USED IN PREDICTIVE ALGORITHM

| Function name | Role of peripherals |
|---|---|
| Grapigcs_open | Initialize LCD screen |
| graphics_get_visible_canvas | Get screen active |
| graphics_fill_canvas | Give color to screen background |

### B. *Discussion*

The synthesis results of the RPC algorithm implementation on the FPGA are very important for the conception to know the occupation rate concerning the internal resources of the FPGA, such as the number of slices, clocks, etc. Table II summarizes the hardware resources used to implement the RPC algorithm on the FPGA. About 50% of the RAM on the FPGA chip is used, and this value can be considered lightly lower compared to the large number of mathematical operations of the genetic predictive algorithm. Besides, 23% of total LUTs (Look-Up Table) with 4 inputs are used. This is important for envisaging other more complex treatments.
In this study, the influence of the population size parameter *n*, the stopping parameter *num* and the prediction horizon $N_y$ on the performances of the RPC controller are investigated.

➢ ***Case 1*:** first, it is required to investigate the influence of $N_y$ on the performance of the RPC. Fig. 6 illustrates the evolution of the output $y(k)$ and the input control $u(k)$ for different values of the prediction horizon $N_y$ ($N_y$=3 and $N_y$=7). These results demonstrate that the output reaches the retained set-point. It can be seen that the response time for $N_y$=7 is higher than that noted in the case where $N_y$=3 (Fig. 6). In fact, when $N_y$=3, the response reaches the point of stability after about 10 iterations but for $N_y$=7 the output stabilizes after 20 samples time. In the second experiment, the aim is to test the influence of the population size parameter *n* and the stopping parameter *num*. Fig. 7 depicts the evolution of the output $y(k)$ and the input control $u(k)$ when *n* is decreased from 300 to 100. Comparing the paces, it can be seen also that when *n* is equal to 300 the FPGA-RPC controller will give a better performance. Two tests are done for the influence of the *num* parameter. First, when the population size parameter is important (*n= 300*), Fig. 8 presents the evolution of system when *num* increases from 10 to 50, it can be noted that between the two pictures there is no big change and it come down to the important value of *n*. Second, when *n* is quite small (*n=100*), Fig. 9 depicts the evolution of the output and the input when the stopping parameter *num* is decreased from 50 to 10, right here the difference is remarkable and it can be seen that the FPGA-RPC controller will produce a better performance when *num* is equal to 50. The choice of the parameter *n* and *num* is for the two genetic algorithms, and the filling of the first population is done randomly.

➢ ***Case 2*:** Fig. 11 presents the evolution of the system for n = 300 and num = 50 when the parameters $a_1$ and $b_1$ are variables. In this example, it is remarkable the effectiveness of the algorithm by seeing that the output follows the set-point and the error is almost null. It is noticeable that the average time required for computing the control input *U* in each sample time is 1*s* for *n*=100 and *num*=50 but also for *n*=300 and *num*=50 is about *3s*. Therefore, it can be concluded that the implementation of the RPC using the GA gives an alternative to control faster systems.

## V. CONCLUSIONS

In this work, linear approach of robust predictive control based on ARIMA model has been implemented into a map Nanoboard 3000XN supported a Xilinx FPGA. To take into account the uncertain dynamic of the process which leads to resolution a non-convex optimization problem, parameters and polytopic uncertainties are considered. A global non-determinist optimization algorithm is proposed to resolve this problem such as the genetic algorithm. The simulation results show that by solving the min-max problem with the proposed optimized approach approve that the output reaches the retained set-point with good performances in terms of pursuit error, rise and response times.

TABLE II
RESOURCES UTILIZATION OF RPC ALGORITHM

| **Used Logic** | **Used** | **Available** | **% of use** |
|---|---|---|---|
| Number of slice flip flops | 2,262 | 22,528 | 10% |
| Number of slices | 3,074 | 11,264 | 27% |
| Number of LUTs (4 inputs) | 4,942 | 22,528 | 21% |
| Total LUTs (4 inputs) | 5,529 | 22,528 | 23% |
| Number of input /output blocks | 145 | 502 | 28% |
| Number of BUFGMUXs | 2 | 24 | 8% |
| Number of RAMB 16BWEs | 16 | 32 | 50% |

## REFERENCES

[1] B. Stellato, T. Geyer and J. G. Paul, *"High-Speed Finite Control Set Model Predictive Control"*, IEEE Transactions on Power Electronics, Mai, Vol. 32, No. 5, pp. 4007-4020, 2017.
[2] B. Yang, T. Yu, H. Shu, J. Dong and L. Jiang , *"Robust sliding –mode control of wind energy conversion systems for optimal power extraction via nonlinear perturbation observers",* Applied Energy, Vol. 210, pp. 711-723, January 2018.
[3] D. E. Goldberg, *"Genetic Algorithm in Search, Optimisation and Machine Learning"*, Addison- Wesley Pub. Co., 1989.
[4] D. Lamburn, P. Gibbens and S. Dumble, *"Explicit efficient constrained model predictive control"*, International Journal of Automation and Control, Vol. 10, No.14, pp. 329 - 355, 2016.
[5] D. Munoz de la Pena, T. Alamo, D. R. Ramırez, E. F. Camacho *, "Min-Max Model Predictive Control as a Quadratic Program",* IET Control Theory & Applications, Vol. 1, No. 1, pp. 328-333, 2007.
[6] D. R. Ramırez, E. F. Camacho and M. R. Arahal, *"Implementation of min-max MPC using hinging hyperplanes application to a heat exchanger",* Control Engineering Practive, Vol. 12, No. 9, pp. 1197-1205, 2004.
[7] E. N. Hartley, J. L. Jerez, A. Suardi, J. M. Maciejowski, E. Kerrigan and G. Constantinides, *"Predictive control using an FPGA with application to aircraft control"*, IEEE Transactions on Control System Technologic, Vol. 22, No. 3, pp. 1006–1017, 2015.
[8] F. Xu, H. Chen, X. Gong and O. Mei, *"Fast Nonlinear Model Predictive Control on FPGA Using Particle Swarm Optimization"*, IEEE Transactions on Industrial Electronics, Vol. 63, No. 1, pp. 310-321, 2016.
[9] J. B. Rawling and D. Q. Mayne, *"Model Predictive Control: theory and design"*, Nob hill publishing, LlC, 2009.
[10] J. J. Grefensette, *"Optimization of control parameters for genetic algorithms"*, IEEE Transaction on Man and Cybernetics, SMC, Vol.16, No. 1, pp. 122-128, 1986.
[11] J. Shaffer, R. Caruana, L. J. Eshelman and R. Des, *"A Study of Control Parameters Affecting Online Performance of Genetic Algorithms for Functions Optimization"*, 3rd International Conference on Genetic Algorithms, Morgan Kaufman, San Mateo, CA, PP.51-60, 1989.
[12] J. Guo and L. Dong, *"Robust load frequency control for uncertain nonlinear interconnected power systems"*, International Journal of Automation and Control, Vol. 11, No. 3, pp. 239-261, 2017.

[13] M. Ricco, P. Manganiello, E. Monmasson, G. Petrone and G. Spagnuolo, *"FPGA-Based Implementation of Dual Kalman Filter for PV MPPT Applications"*, IEEE Transactions on Industrial Informatics, Vol. 13, No. 1, pp. 176-185, 2017.

[14] S. P M. Alinodehi, S. J. Louis, S. Moshfe and M. Nicolescu, *"A Modified Steady State Genetic Algorithm Suitable for Fast Pipelined Hardware"*, IEEE Congress on Evolutionary Computation, 5-6 June, Sain sebastian, Spain, 2017.

[15] S. Lucia, D. Navarro, O. Lucia, P. Zometa and R. Findeisen, *"Optimized FPGA Implementation of Model Predictive Control for Embedded Systems Using High Level Synthesis Tool"*, IEEE Transactions on Industrial Informatics, Vol. 14, No. 1, pp.137-145, 2017.

[16] T. S. Brini, B. Bouzouita and F. Bouani, *"FPGA Implementation of a Model Predictive Control: Application to a control system of water level",* The 3rd International Conference on Automation, Control Engineering and Computer Science (ACECS), March 20-22, Hammamet, Tunisia**,** 2016.

[17] X. J. Wu, X. J. Zhu, G.-Y. Cao and H.-Y. Tu, *"Predictive control of sofc based on a ga-rbf neural network model",* Journal of Power Sources, Vol. 179, No. 1, pp. 232-239, 2007.

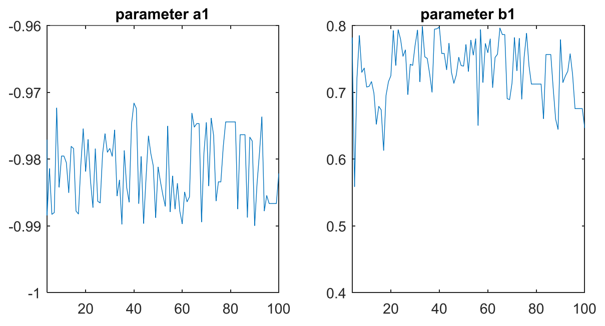

Fig. 4 case 1: Evolution of model estimated parameters

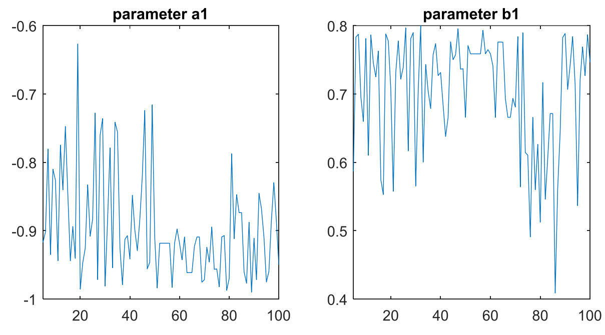

Fig. 5 case 2: Evolution of model estimated parameters

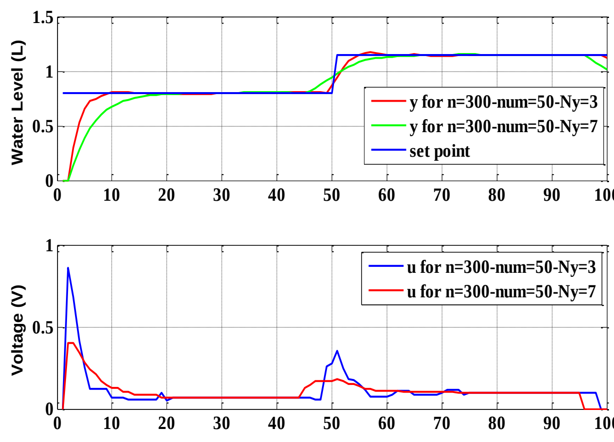

Fig. 6 case 1: Evolution of the output y and the control input u for n=300, num=50, Ny=7 and Ny=3

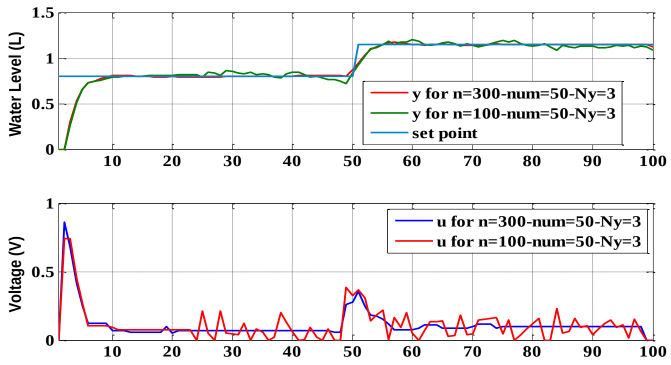

Fig. 7 case 1: Evolution of the output y and the control input u for n=300 and n=100, num=50, Ny=3

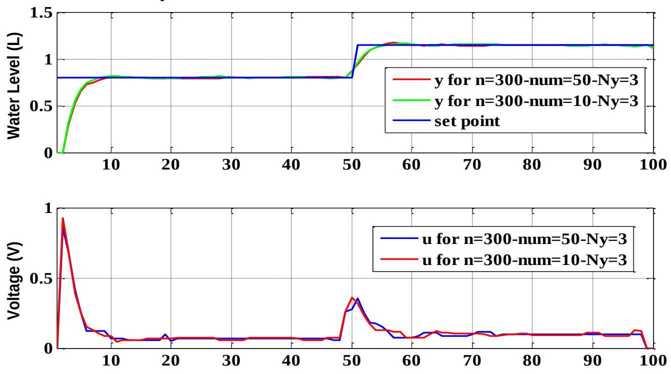

Fig. 8 case 1: Evolution of the output y and the control input u for n=300, num=50 and num=10, Ny=3

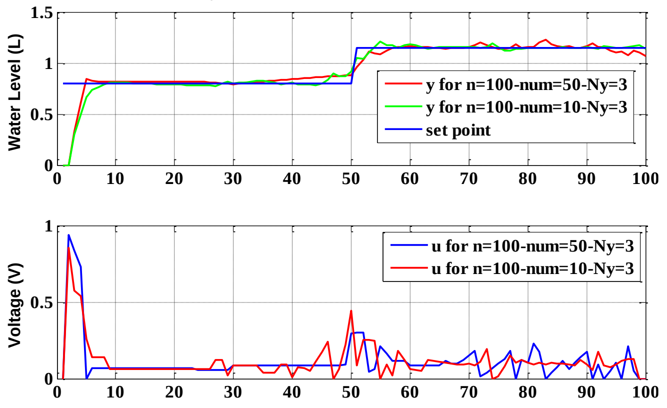

Fig. 9 case 1: Evolution of the output y and the control input u for n=100, num=50 and num=10, Ny=3

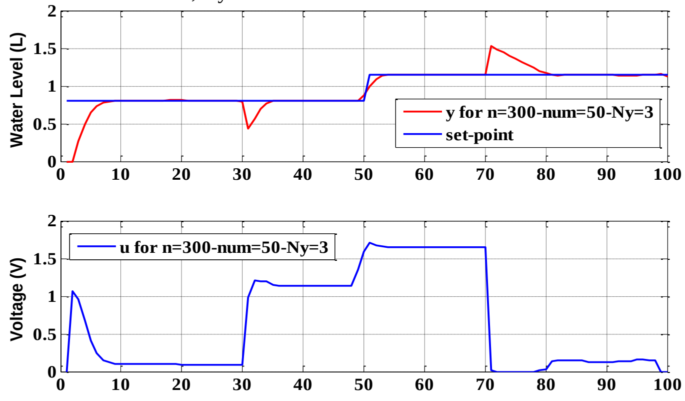

Fig. 10 case 2: Evolution of the output y and the control input u for n=300, num=50, Ny=3